OSSに対する不完全デバッグを考慮した確率微分方程式モデルと最適バージョンアップ問題に関する一考察 (不確実・不確定性下での意思決定過程)

AUTHOR(S):

田中, 智朗; 田村, 慶信; 山田, 茂

CITATION:

田中, 智朗 ...[et al]. OSSに対する不完全デバッグを考慮した確率微分方程式モデルと最適バージョンアップ問題に関する一考察 (不確実・不確定性下での意思決定過程). 数理解析研究所講究録 2010, 1682: 217-224

ISSUE DATE:

2010-04

URL:

http://hdl.handle.net/2433/141381

数理解析研究所講究録
第1682巻 2010年 217-224

OSSに対する不完全デバッグを考慮した確率微分方程式モデルと最適バージョンアップ問題に関する一考察

鳥取大学大学院・工学研究科　田中 智朗 (Tomoaki Tanaka)[†]
山口大学大学院・理工学研究科　田村 慶信 (Yoshinobu Tamura)[††]
鳥取大学大学院・工学研究科　山田　茂 (Shigeru Yamada)[†]

[†]Graduate School of Engineering, Tottori University
[††]Graduate School of Science and Engineering, Yamaguchi University

1　まえがき

近年では，インターネットの普及により世界中で同時に新しい情報を得ることができるようになり，実時間的あるいはインタラクティブ性の高い機能の追求へとコンピュータに対する関心が深まっているといえる．このような状況から，ネットワークを基本にしたソフトウェアの分散開発，およびソフトウェアそのものの分散化がさらに拡大してきた．現在の開発環境は，作業効率の改善に限界がきた大型ホストコンピュータ中心の集中型開発環境から開発者が互いに離れていてもネットワークで同様の作業ができるように，多数のワークスステーションを相互接続した分散開発環境に変わりつつある．[1,2,3]

本論文では，オープンソースソフトウェア（Open Source Software，以下OSSと略す）の開発環境下において，フォールト発見事象は確率的変動を伴い，不規則な振る舞いを示すと考えられることから，ソフトウェア故障強度に対して不規則性を導入することにより確率微分方程式に基づくソフトウェア信頼度成長モデル（Software Reliability Growth Model，以下SRGMと略す）を構築する．これにより，複雑なOSSの開発状況を考慮した信頼性評価が可能になると考える．また，OSSのリリース候補版において，十分な信頼性を確保することは，正式版リリース後のユーザに対する信頼に大きく関わるだけでなく，OSSの保守労力の増大にも大きく関係する．したがって，最適なバージョンアップ時期を推定することは，OSS開発において重要な段階となる．さらに，OSSの開発環境においては，開発者だけでなくユーザもフォールト報告を行ったり，開発経験の浅いユーザやソフトウェア開発者も自由にソースコードを書き換えられることができるなど，通常の企業や組織が開発する環境とは大きく異なる点がある．このことから，本論文では，不完全デバッグ発生確率を考慮した最適バージョンアップ時刻を推定する．

2　確率微分方程式モデル

時刻$t=0$でOSSがリリースされ，任意の時刻tにおけるソフトウェア内の発見フォールト数$S(t),(t\geq 0)$は，以下の常微分方程式によって記述されるものと仮定する．

$$\frac{dS(t)}{dt}=\lambda(t)S(t). \tag{1}$$

ここで，$\lambda(t)(>0)$は時刻tにおけるソフトウェア故障強度を表す．

また，OSSのフォールト発見事象は常に不規則であると考えられる．さらに，OSSは常にバグフィックスやコンポーネントの加除が繰り返されており，ソフトウェア故障強度もそれらに応じて変化するものと考えられる．したがって，式(1)のソフトウェア故障強度$\lambda(t)$に不規則性を導入すると，式(1)は，

$$\frac{dS(t)}{dt}=\{\lambda(t)+\sigma\gamma(t)\}S(t), \tag{2}$$

となる．ここで$\sigma(>0)$は定数パラメータであり，$\gamma(t)$は標準化されたGauss型白色雑音である．式(2)を，以下のItô型[4,5]の確率微分方程式に拡張して考える．

$$dS(t)=\left\{\lambda(t)+\frac{1}{2}\sigma^2\right\}S(t)dt+\sigma S(t)dW(t). \tag{3}$$

式 (3) で定義された $W(t)$ は Wiener 過程であり，白色雑音の形式的な時間積分，

$$W(t) = \int_0^t \gamma(s)ds, \tag{4}$$

として表すことができる．また，ソフトウェア故障強度関数 $\lambda(t)$ は次の指数形関数と S 字形関数の 2 種類を仮定する．

$$\int_0^t \lambda(s)ds = \sum_{i=1}^{n} p_i(1-\exp[-\alpha_i t]), \tag{5}$$

$$\int_0^t \lambda(s)ds = \sum_{i=1}^{n} p_i\{1-(1+\alpha_i t)\exp[-\alpha_i t]\}. \tag{6}$$

ここで，α_i は各ソフトウェアコンポーネントのソフトウェア故障強度の加速係数，p_i は各コンポーネントの開発規模，n はコンポーネントの数を表す．

OSS の開発では，常にフォールト修正やバージョンアップなどが繰り返されており，使用頻度や人気の高い OSS になるほどフォールト報告が頻繁に行われている．この運用形態は OSS の開発プロジェクトが無意味なものとみなされて解散されるまで続けられる．したがって，1 つの企業組織内において，ある特定の使用目的に限定されたソフトウェアの開発を対象としている従来の SRGM により，OSS の信頼度成長現象を包括することは難しくなる [6]．そのため，本論文では，

$$\lim_{t\to\infty} \mathrm{E}[S(t)] = \infty, \tag{7}$$

を満たす確率微分方程式モデルを適用する．ここで，$\mathrm{E}[S(t)]$ は任意の時刻 t における発見フォールト数の期待値を表す．

3 ソフトウェア信頼性評価尺度

3.1 発見フォールト数の期待値と分散

ソフトウェアの信頼性評価尺度を導出するにあたり，まずは任意の時刻 t における発見フォールト数の期待値 $\mathrm{E}[S(t)]$ を求める．

Wiener 過程 W(t) の密度関数が

$$f(W(t)) = \frac{1}{\sqrt{2\pi t}}\exp\left\{-\frac{W(t)^2}{2t}\right\}, \tag{8}$$

であることから，任意の時刻 t までに発見される総フォールト数の期待値 $\mathrm{E}[S(t)]$ は，

$$\begin{aligned}\mathrm{E}[S(t)] &= \mathrm{E}\left[v\cdot\exp\left(\int_0^t \lambda(s)ds + \sigma W(t)\right)\right] \\ &= v\cdot\exp\left(\int_0^t \lambda(s)ds + \frac{\sigma^2}{2}t\right),\end{aligned} \tag{9}$$

となる．また，任意の時刻 t までに発見される総フォールト数の分散は，

$$\begin{aligned}\mathrm{Var}[S(t)] &\equiv \mathrm{E}[\{S(t)-\mathrm{E}[S(t)]\}^2] \\ &= v^2\cdot\exp\left(2\int_0^t \lambda(s)ds + \sigma^2 t\right)\cdot\{\exp(\sigma^2 t)-1\},\end{aligned} \tag{10}$$

となる．

3.2 平均ソフトウェア故障発生時間間隔

- 瞬間 MTBF

任意の時刻 t における瞬間的なフォールト発見間隔の平均を意味する瞬間 MTBF（以下，$MTBF_I$ と略す）を導出する．本論文では，$MTBF_I$ を簡略化のために，

$$MTBF_I(t) = \frac{dt}{\mathrm{E}[dS(t)]}, \tag{11}$$

で近似的に計算する．これにより本モデルの $MTBF_I(t)$ は，

$$MTBF_I(t) = 1 \Big/ \left\{ v\left(\lambda(t) + \frac{1}{2}\sigma^2\right) \cdot \exp\left(\int_0^t \lambda(s)ds + \frac{\sigma^2}{2}t\right)\right\}, \tag{12}$$

となる．

- 累積 MTBF

運用開始時点から考えたときの発見フォールト 1 個当りに要する発見時間の平均を意味する累積 MTBF（以下，$MTBF_C$ と略す）を導出する．瞬間 MTBF と同様に，本論文では $MTBF_C$ を簡略化のために，

$$MTBF_C(t) = \frac{t}{\mathrm{E}[S(t)]}, \tag{13}$$

と近似的に計算する．これにより本モデルの $MTBF_C(t)$ は，

$$MTBF_C(t) = t \Big/ \left\{ v \cdot \exp\left(\int_0^t \lambda(s)ds + \frac{\sigma^2}{2}t\right)\right\}, \tag{14}$$

となる．

4 モデルの適合性評価

適用したモデルの実測データに対する適合性比較を行う．ここで適合性尺度としては，平均偏差二乗和 (Mean Square Error, 以下 MSE と略す) を用いる．MSE は実測値と推定値との二乗誤差をデータ数で平均化したものである．ここで，一定のテスト時刻 t_k までに発生した累積フォールト数 y_k に関する K 組の発見フォールト数データ $(t_k, y_k)(k = 1, 2, \cdots, K)$ が観測されているものとすると，

$$MSE = \frac{1}{K}\sum_{k=1}^{K}(y_k - \hat{y}_k)^2, \tag{15}$$

により算出される．このとき $\hat{y}_k$ は，テスト時刻が $t_k(k = 1, 2, \cdots, K)$ のときの推定値である．MSE は，その値が小さいほど実測データによく適合していることを意味する．

5 最適バージョンアップ時刻の推定

ソフトウェア製品の運用段階における品質は，実施されるテストの質と量および総テスト時間に依存する．つまり，ソフトウェアの高信頼性を実現させるために，テスト時間を増加させれば，ソフトウェア内に潜在する多くのフォールトを発見および除去することが可能であり，運用段階におけるソフトウェアの信頼性は向上する．しかしながら，長時間テストを実施すればするほど，その分ソフトウェアの信頼性は向上するが，テストに費やすコストが増加する．逆に，テストの実施時間を減少させると，テストに費やすコストは減少し，ソフトウェア製品の出荷も早くなるが，ソフトウェア内の残存フォールト数が大きくなり，運用段階での保守コストが増大することになる．ここでは，総期待ソフトウェアコストを評価基準として，一般化された確率微分方程式モデルに基づいてソフトウェアの最適バージョンアップ問題について定式化する．

まず，OSS の開発に伴う開発労力は，

$$E_1(t) = m_0 \cdot \mathrm{E}[S(t)], \tag{16}$$

と表される．ここで，m_0 はフォールト 1 個当りの修正労力を表す．また，メジャーバージョンアップ後の保守労力は，

$$E_2(t) = m_1\{\mathrm{E}[S(t_0)] - \mathrm{E}[S(t)]\} + m_2 t, \tag{17}$$

となる．ここで，m_1 はフォールト 1 個当りの保守労力，m_2 は単位時間当りの保守労力，t_0 は前回のバージョンアップ時期を表す．また，時間区間 $(0, t_0]$ を超えてコンポーネントを新規に開発する際には，整合性を確認するためのペナルティ関数を課されるものとし，ペナルティ関数を以下のように定義する．

$$G(t) = (1-c)\exp\left[\frac{t-t_0}{u}\right]. \tag{18}$$

ここで，c はシステム全体に対する新規コンポーネントの割合，u は過去のメジャーバージョンアップ回数を表す．

したがって，総期待開発労力は，

$$E(t) = E_1(t) + E_2(t) + G(t), \tag{19}$$

となる．式 (19) を最小にする時刻 t^* が，OSS の最適バージョンアップ時刻となる．

6 不完全デバッグの発生確率

OSS の開発環境においては，開発者だけでなくユーザがフォールト報告を行ったり，また開発経験の浅いユーザやソフトウェア開発者も自由にソースコードを書き換えられることができるなど，通常の企業や組織が開発する環境とは大きく異なる点がある．そこで，不完全デバッグの発生確率を評価尺度として用いて最適バージョンアップ時期推定を行うことは有用であると考えられる．

テスト時刻 $t(t \geq 0)$ までに発見されたフォールト数 $S(t)$ が x である条件の下で，微小時間区間 $\Delta t(t > 0)$ 後において発見フォールト数が増加しない確率を，ここでは不完全デバッグ事象の発生確率とする．また，不完全デバッグの発生確率は，

$$\Pr[S(t+\Delta t) \leq x | S(t) = x] = \Phi\left[\left\{\log\frac{\int_0^{t+\Delta t}\lambda(s)ds}{\int_0^t \lambda(s)ds}\right\} \Big/ \sigma\sqrt{\Delta t}\right], \tag{20}$$

となる．ここで，$\Pr[\cdot]$ は確率を表し，$\Phi(\cdot)$ は標準正規分布関数であり，

$$\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x}\exp\left(-\frac{z^2}{2}\right)dz, \tag{21}$$

と定義される．

7 数値例

7.1 ソフトウェア信頼性評価尺度

数値例として，本論文では提案された確率微分方程式モデルの適用例を示すために，Firefox 2 RC1 版 [7] を取り上げる．Firefox 2 RC1 において，前回のリリース時点における発見フォールト数は 216 であることから，$v = 216$ とおき，未知パラメータ α_i と σ を最小二乗法により推定した．指数形故障強度関数および S 字形故障強度関数を適用した推定結果を表 1 および表 2 に示す．

次に，Firefox2 RC1 に対する信頼性評価の一例を示す．Firefox 2 RC1 において，式 (9) における累積発見フォールト数の期待値の推定値 $\hat{\mathrm{E}}[S(t)]$ を図 1 に示す．さらに式 (14) における $MTBF_C$ の推定値 $\hat{MTBF}_C(t)$ を図 2 に示す．図 2 のように $MTBF_C$ が大きな値をとることは，ソフトウェアの信頼性が向上していることを意味する．また，式 (15) における MSE によるモデルの適合性評価結果を示す．指数形故障強度関数の場合の MSE は，536.68 となり，S 字形故障強度関数の場合の MSE は 417.10 となった．MSE の値が S 字形の方が小さい値をとることから，指数形のものより，S 字形の方が適合性が高いことが確認できる．

表 1： Firefox2 RC1 リリース時の指数形ソフトウェア故障強度関数におけるパラメータの推定結果.

Model Parameters	Estimated Values
α_1	0.012585
α_2	0.0122959
α_3	0.0119231
α_4	0.0126214
α_5	0.0119231
α_6	0.0122959
α_7	0.0119231
α_8	0.0119231
α_9	0.0122959
α_{10}	0.0119231
α_{11}	0.012585
α_{12}	0.0126374
α_{13}	0.0119231
α_{14}	0.0126385
α_{15}	0.0112344
α_{16}	0.0126374
α_{17}	0.0124901
σ	0.109762

表 2： Firefox2 RC1 リリース時の S 字形ソフトウェア故障強度関数におけるパラメータの推定結果.

Model Parameters	Estimated Values
α_1	0.0264652
α_2	0.0264684
α_3	0.0262256
α_4	0.0264162
α_5	0.0262256
α_6	0.0264684
α_7	0.0262256
α_8	0.0262256
α_9	0.0264684
α_{10}	0.0262256
α_{11}	0.0264652
α_{12}	0.0643793
α_{13}	0.0262256
α_{14}	0.0264358
α_{15}	0.0250269
α_{16}	0.0643793
α_{17}	0.0264856
σ	0.10275

7.2 最適バージョンアップ時期推定

ここでは，不完全デバッグ発生確率を考慮しない場合における最適バージョンアップ時期推定を行う．最適バージョンアップ時期推定の実行例として，まず，前回の開発開始からバージョンアップまでの開発期間 t_0 を調べると，開発期間が 137 日間であったことから，$t_0 = 137$ と仮定する．パラメータが $(t_0 = 137, c = 0, u = 1, m_0 = 1.0, m_1 = 2.28, m_2 = 1.28)$ で与えられたときの式 (19) の推定された総期待開発労力を図 4 に示す．推定された最適バージョンアップ時期と総期待開発労力の推定値は指数形故障強度関数のものは，最適バージョンアップ時刻は開発開始から 143 日後であり，そのときの総期待開発労力は 1278.75（人 · 日）であった．また，S 字形故障強度関数の最適バージョンアップ時刻は開発開始から 143 日後であり，そのときの総期待開発労力は 1280.43（人 · 日）となった．

7.3 不完全デバッグ発生確率を考慮した最適バージョンアップ時刻の推定

Firefox 2 RC1 の不完全デバッグ発生確率を考慮したバージョンアップ時刻を推定するために，Alpha 版第 2 版がバージョンアップされた時点における最適バージョンアップと総期待開発労力を推定する．ここで推定された最適バージョンアップ時刻と総期待開発労力を図 3 に示す．図 3 より，Alpha 版第 2 版における最適バージョンアップ時刻は指数形故障強度関数の場合においては，開発開始から 53 日後であり，そのときの総期待開発労力は 286.21（人 · 日）であった．また同様にして，S 字形故障強度関数の場合においては，最適バージョンアップ時刻は開発開始から 53 日後であり，その時の総期待開発労力は 1280.43（人 · 日）であった．

図 5 および図 6 は Alpha 版第 2 版がバージョンアップされた時点における不完全デバッグ発生確率を表している．ここで微小時間 $\Delta t = 5$ と仮定すると，先ほどの図 3 より Alpha 版第 2 版において指数形故障強度関数，S 字形故障強度関数共に最適バージョンアップ時刻は開発開始から 53 日後であると推定することができた．そこで Alpha 版第 2 版が開発開始から 53 日後にバージョンアップされた場合における不完全デバッグ発生確率を求め，その値を Firefox 2 RC1 がバージョンアップされたときにおける目標不完全デバッグ発生確率とする．具体的には，Alpha 版第 2 版が 53 日後にバージョンアップされたときの不完全デバッグ発生確率は指数形故障強度関数，S 字形故障強度関数共に 0.3990 であることから，Firefox2 RC1 における目標不完全デバッグ発生確率を 0.3990 とし，Firefox 2 RC1 における不完全デバッグ発生確率 0.3990 に達した時点が不完全デバッグ発生確率を考慮した最適バージョンアップ時刻であるとする．また，Firefox 2 RC1 における不完全デバッグ発生確率として，指数形故障強度関数のものを図 7 に示す．さらに，S 字形故障強度関数のものを図 8 に示す．

表 3： Alpha 版第 2 版リリース時の指数形ソフトウェア故障強度関数におけるパラメータの推定結果.

Model Parameters	Estimated Values
α_1	0.577586
α_2	0.551347
α_3	0.531666
α_4	0.614390
α_5	0.531666
α_6	0.551347
α_7	0.531666
α_8	0.531666
α_9	0.551347
α_{10}	0.531666
α_{11}	0.577586
α_{12}	0.602286
α_{13}	0.531666
α_{14}	0.595216
α_{15}	0.502565
α_{16}	0.602286
α_{17}	0.566008
σ	0.333165

表 4： Alpha 版第 2 版リリース時の S 字形ソフトウェア故障強度関数におけるパラメータの推定結果.

Model Parameters	Estimated Values
α_1	0.723020
α_2	0.787713
α_3	0.801933
α_4	0.527040
α_5	0.801933
α_6	0.787713
α_7	0.801933
α_8	0.801933
α_9	0.787713
α_{10}	0.801933
α_{11}	0.723020
α_{12}	0.605919
α_{13}	0.801933
α_{14}	0.645583
α_{15}	0.769220
α_{16}	0.605919
α_{17}	0.758429
σ	0.329018

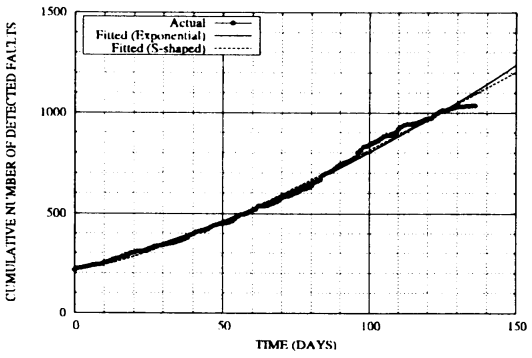

図 1： Firefox 2 RC1 における累積発見フォールト数.

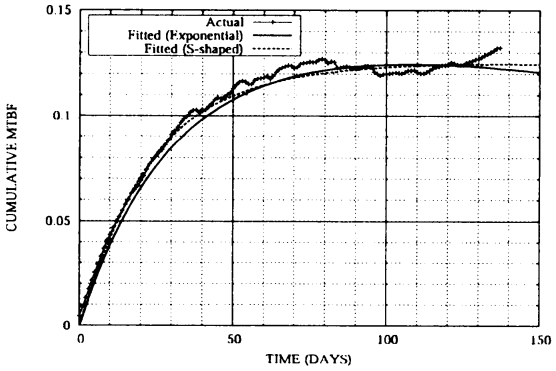

図 2： Firefox 2 RC1 における累積 MTBF.

図 9 は，指数形故障強度関数における不完全デバッグ発生確率を考慮した最適バージョンアップ時刻の推定結果とそのときの総期待開発労力を表している．ここで目標不完全デバッグ発生確率が 0.3990 であることから，図 9 に示される不完全デバッグ発生確率が 0.3990 に達した時点が不完全デバッグ発生確率を考慮した最適バージョンアップ時刻となる．この図において，不完全デバッグ発生確率が 0.3990 に達する時点は開発開始から 118 日後であり，そのときの総期待開発労力は 1633.93（人 · 日）となることなる．不完全デバッグ発生確率を考慮した場合と，不完全デバッグ発生確率を考慮しない場合とを比べると開発期間は短くなり，総期待開発労力は大きくなる．S 字形故障強度関数においては，図 10 から不完全デバッグ発生確率が 0.3990 に達する時点は開発開始から 112 日後であり，そのときの総期待開発労力は 1660.22（人 · 日）となる．S 字形故障強度関数においても不完全デバッグ発生確率を考慮した場合と，不完全デバッグ発生確率を考慮しない場合を比較すると開発期間は短くなり，総期待開発労力は大きくなる様子が確認できる．

8 むすび

本論文では，OSS のフォールト発見事象が常に不規則な状態であると考え，OSS の特殊な開発形態を考慮するため，Itô 型確率微分方程式に基づいた SRGM を提案した．また数値例として，バグトラッキングシステム上にある Firefox のフォールトデータを用いて，Alpha 版第 2 版の最適リリース時刻における不完全デバッグ発生確率を求め，その推定値を Firefox 2 RC1 での目標不完全デバッグ発生確率とすることによ

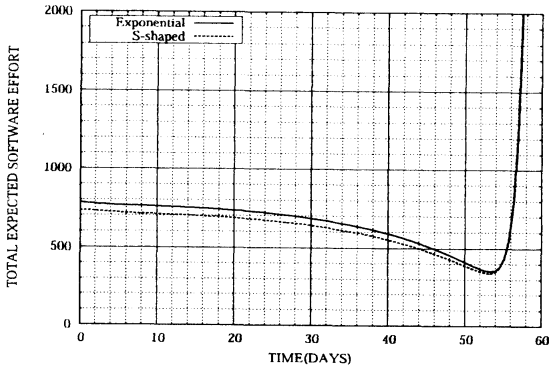

図 3： Alpha 版第 2 版における推定された最適バージョンアップ時期と総期待開発労力.

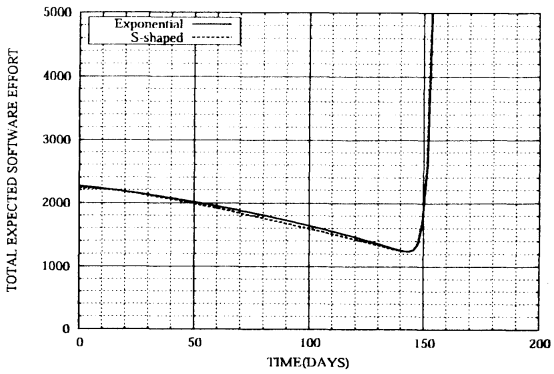

図 4： Firefox 2 RC1 における推定された最適バージョンアップ時期と総期待開発労力.

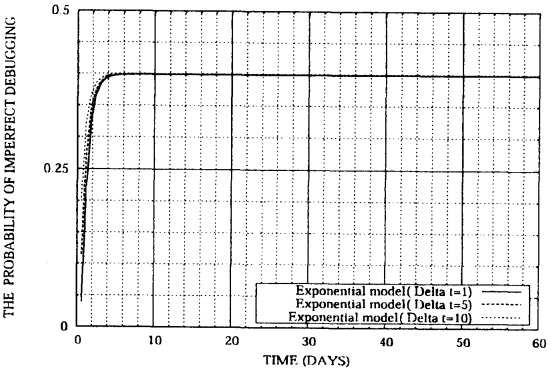

図 5： Alpha 版第 2 版の指数形強度関数における不完全デバッグ発生確率.

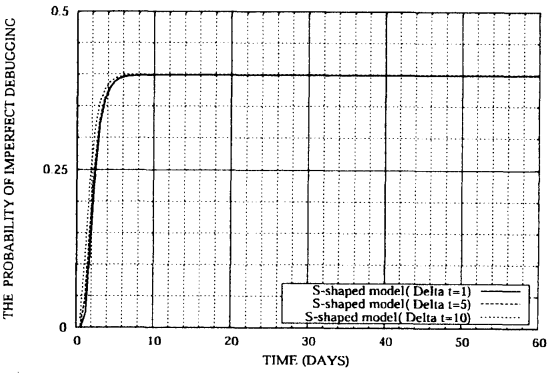

図 6： Alpha 版第 2 版の S 字形強度関数における不完全デバッグ発生確率.

り，不完全デバッグ発生確率を考慮した最適バージョンアップ時刻を推定した．これにより，より現実的なバージョンアップ時期推定を行うことができることを示した．今後もオープンソースプロジェクトに基づく開発形態は急速に発展するものと考えられることから，このような OSS を開発するオープンソースプロジェクトの下で不完全デバッグ発生確率を考慮した最適バージョンアップ時刻の推定は，信頼性評価法として利用できるものと考えられる．

謝辞

本論文の一部は，文部科学省科学研究費若手研究 (B)（課題番号 21700044）の援助を受けたことを付記する．

参考文献

[1] 松本正雄，小山田正史，松尾谷徹，ソフトウェア開発検証技師，電子情報通信学会，東京，1997.

[2] R.S.Pressman, *Software Engineering: A practitioner's Approach (4th ed.)*, McGraw-Hill, New York,1982.

[3] A. Umar, *Distributed Computing and Client-Sever System,* Prentice Hall, New Jersey, 1993.

[4] L. Arnold, *Stochastic Differential Equations-Theory and Applications,* John Wiley and Sons, New York, 1974.

[5] E. Wong, *Stochastic Processes in Information and Systems,* McGraw-Hill, New York, 1971.

[6] 田村慶信，肌附康司，山田茂，木村光宏，"オープンソースソフトウェアに対するユーザ指向の信頼性評価ツールの開発," Linux Conference, 東京, 2006 年 5 月 31 日-6 月 2 日, http://lc.linux.or.jp/lc2006/

[7] Mozilla,org, Mozilla Foundation. [Online]. Avail-able: http://www.mozilla.org/

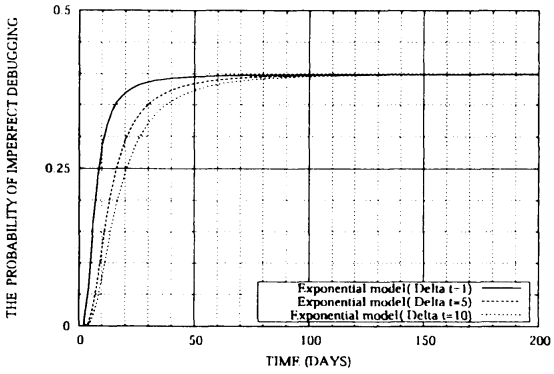

図 7： Firefox 2 RC1 の指数形強度関数における不完全デバッグ発生確率.

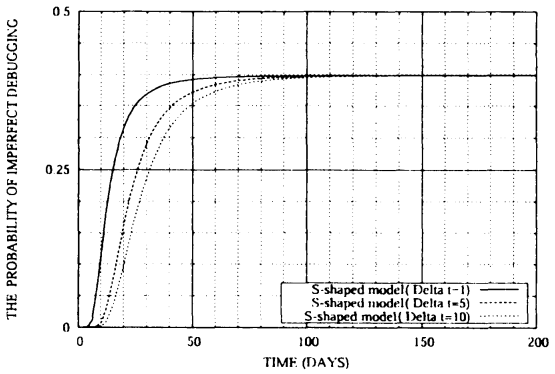

図 8： Firefox 2 RC1 の S 字形強度関数における不完全デバッグ発生確率.

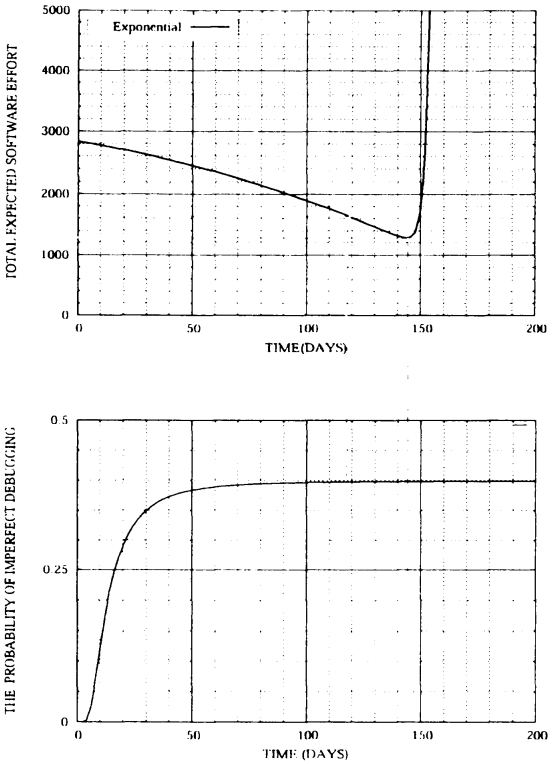

図 9： 指数形強度関数における不完全デバッグ発生確率を考慮した最適バージョンアップ時期と総期待開発労力.

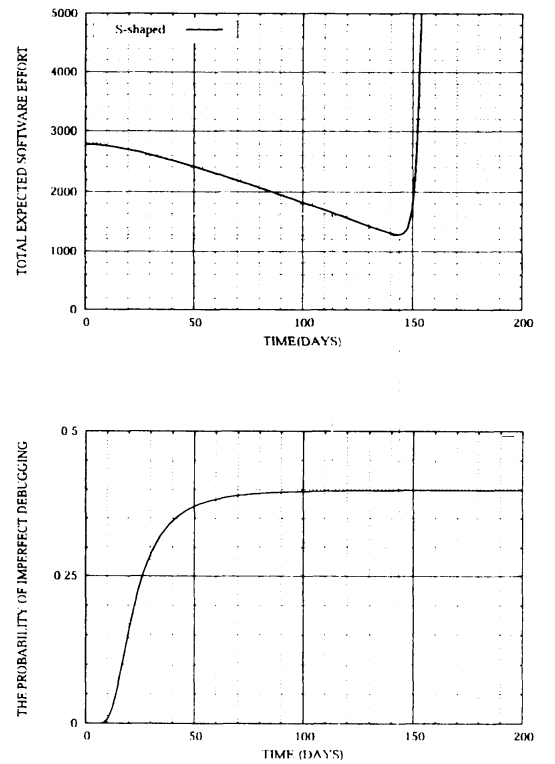

図 10： S 字形強度関数における不完全デバッグ発生確率を考慮した最適バージョンアップ時期と総期待開発労力.